**SOFTWARE DESIGN SPECIFICATION**

1. **Introduction**
   1. **Purpose**

This document's purpose is to provide a high-level design framework around which to build our project Medwle.  It also provides a list of requirements against which to test the final project and determine whether we were able to successfully implement the system according to design. It also focuses on specifying a high-level view of the architecture of our system, and on the interaction between the user and the system.

* 1. **Scope of the Development of the Project**

*In scope:*

1. Managing Patient’s Profile, which would include particulars of a patient, his/her appointment records, and prescriptions.
2. Managing a Doctor’s profile, which would include particulars of a doctor, details about his/her patients, appointments.
3. Giving alerts to patients for their booked appointments.
4. Look up facility for Nearby Homeopathic Medicine stores.

*Out of scope:*  
1. Make Medwle platform independent since presently its only desired to work on Android Phones.

* 1. **Acronyms and Abbreviations**

1. Medwle: Medwle , name of our Android Application
2. SRS: Software Requirements Specification
3. GUI: Graphical User Interface
   1. **References**

N/A

1. **Conceptual Architecture/Architecture Diagram**
   1. **Overview of Module/Components**

Our system is designed with extensibility and scalability in mind.  We are taking great care in designing the Application. Many of the anticipated changes to our system in future phases will only require adding new types of data and changing the user presentation code to make use of these new data. The framework we have designed will only require "plugging in" these new structures without refactoring the major logic.

**2.2 Structure and Relationships**

* Login/Sign-up
  + Login Authentication (Must be a registered user)
  + Sign Up and create a new account if a new user(doctor or patient)
  + Using features like book appointment, create prescription, edit prescription etc.
  + Implemented in java and Android.
* Booking and cancelling appointments
  + Patient logs in using a registered account
  + Patient searches for doctors, and chooses a doctor according to their need.
  + Can cancel an existing appointment.
  + Input appointment details like time, date etc and save the appointment
  + Implemented in java and Android.
* Creating and editing Prescription
  + Doctor logs in using a registered account.
  + Can view patients profile, prescriptions etc.
  + Add new prescriptions or edit existing ones.
  1. **User Interface Issues**

The user interface of the Medwle will be divided into five main sections: The “Client choosing Section”, The "Login Authentication or Sign-up”, the "Appointment booking”, the "Prescription Creating/Editing” and the "Profile”. These tools will allow the various users to accomplish the management of the software, and the management of the rest of their functionalities.  The personas described in the SRS represent the types of people who will make use of the Medwle - and the purpose of this section is to describe how the user interfaces allow those people to do their tasks.

**2.3.1 Login Authentication**

This application will reside on an Android Phone, and will be used mostly by Patient, Doctor.

The login/signup will have the following basic layout:

**Login Screen -**This first screen will allow *patients and doctors* to protect rest of the application by using email ID and password verification.

**Sign-up Screen -**This screen allows *patients and Doctors* to create a new ID and use the features of our application.

**2.3.2 Appointment Booking/Cancelling**

This application will reside on the Medwle for only Patient Login. It will have the following layout:

**Details Fill Up Screen –**This will allow the patient to create a new appointment.

* Fill up the necessary details.
* Choose a doctor.
* Book an appointment.
* Get Notified few hours before the appointment.

**2.3.2 Appointment Cancelling**

This application will reside on the Medwle for only Patient Login. It will have the following layout:

**Details Fill Up Screen –**This will allow the patient to cancel an already existing appointment.

* Fill up the necessary details.
* Cancel the appointment.

**2.3.3 Creating Prescription/ Editing an existing one**

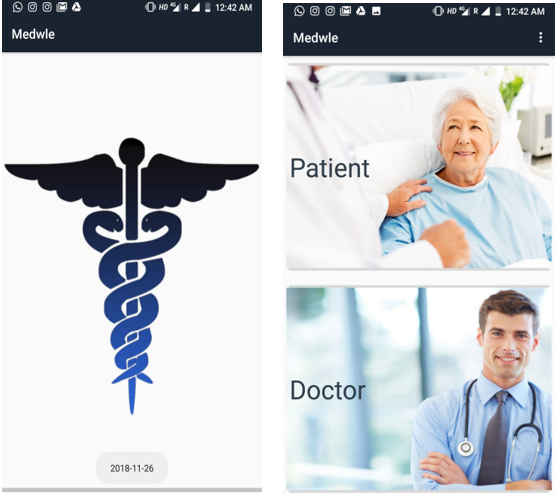
This application will reside on the Medwle for only Doctor Login. It will have the following layout:

**Create a new prescription –**Doctor can view a patient’s profile and add new prescriptions to it. They will need to fill up basic details and write down the prescription. The prescription will be saved.

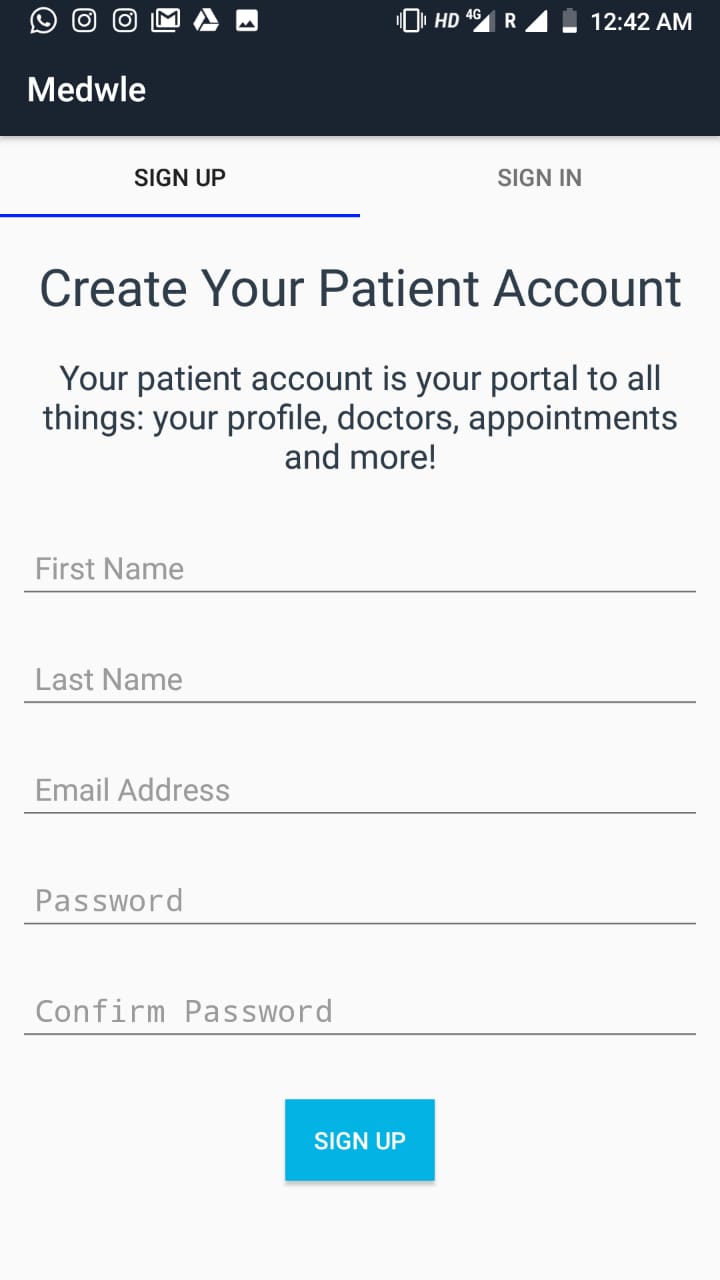
**Editing an existing prescription –**Doctor can view a patient’s profile and prescriptions in it. They will need to fill up basic details and edit the prescription. The edited prescription will be saved.

**2.3.4 Screenshots:**

1. Loading Screen & First Activity

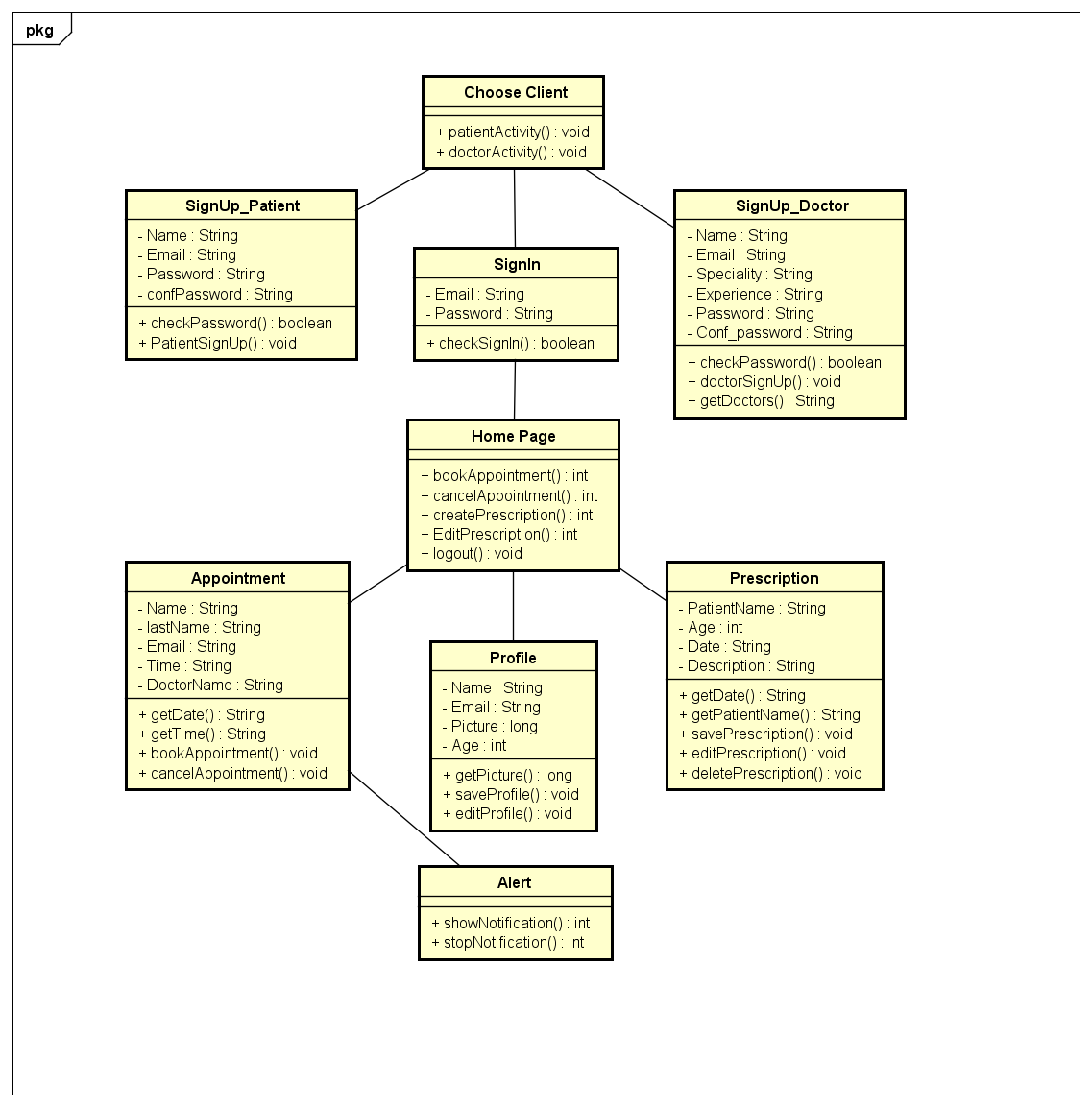


2.) Sign Up and Sign In For Doctor as well as Patient:

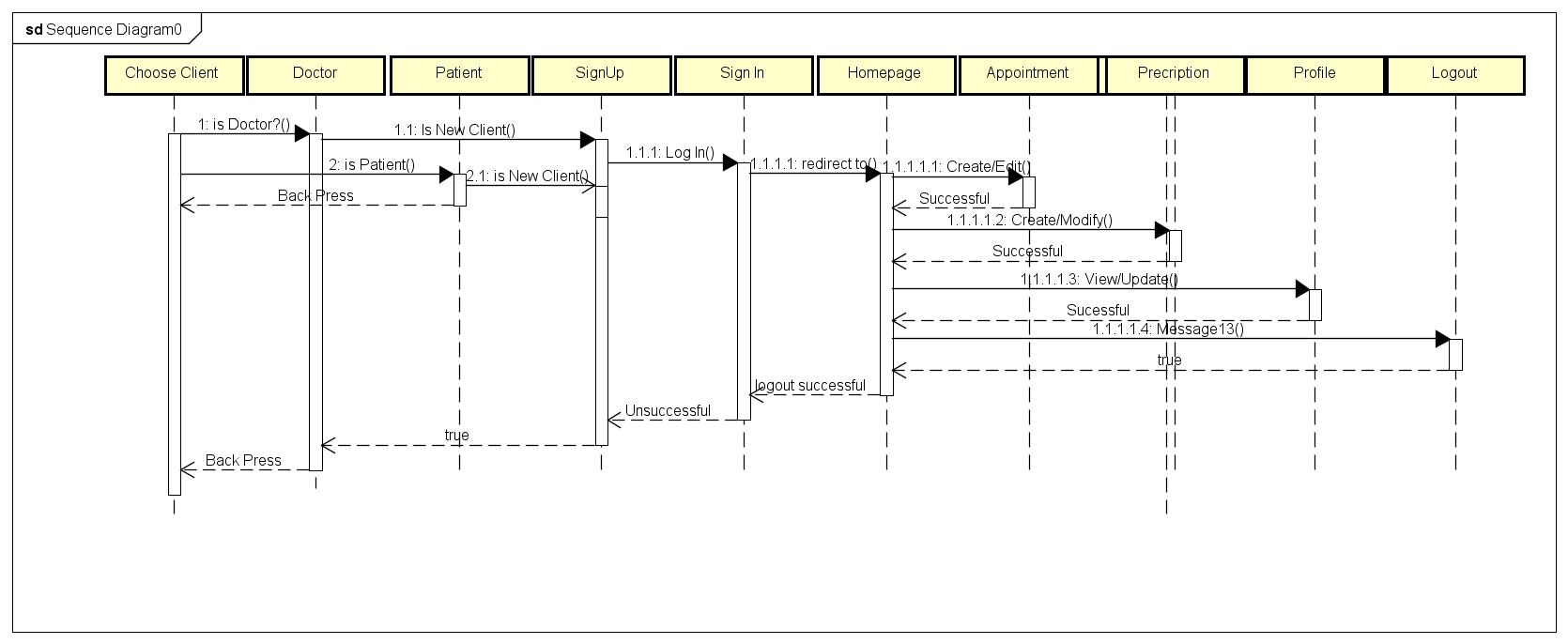


## **3.Logical Architecture**

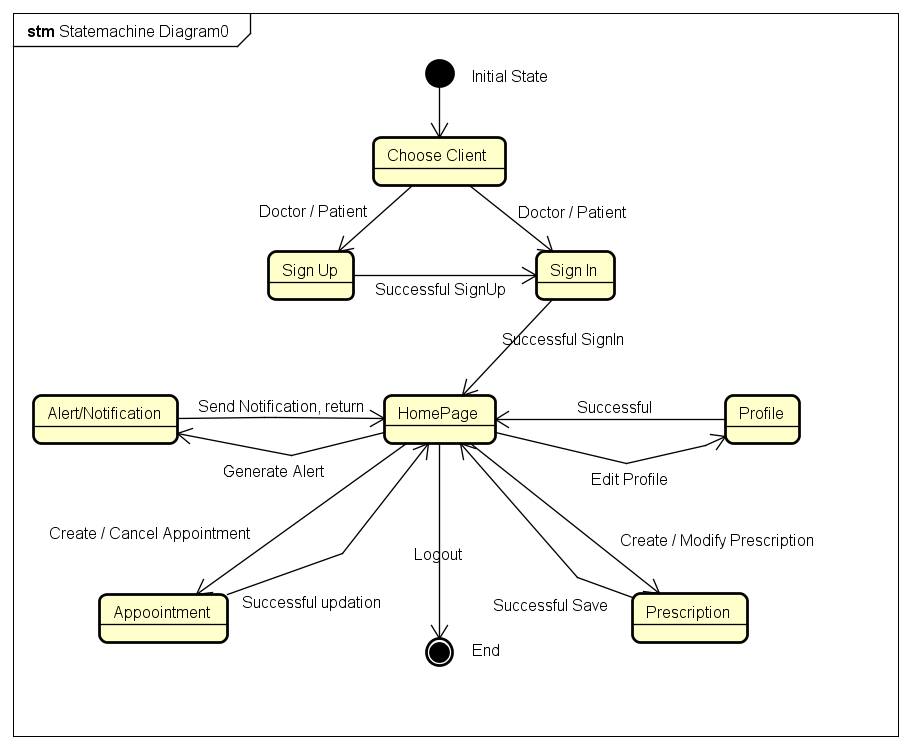
* 1. ***Class Diagram***



* 1. ***Sequence Diagram***



* 1. ***State Diagram***



## **5.0 Design decisions and trade-offs**

### **5.1 Kotlin/Java Debate**

The first issue we dealt with as a group was whether to develop in Java or Kotlin. Two of our members wanted to develop in Kotlin because they are familiar with it and enjoy working with it more. Kotlin is an upcoming language which can be used for android development. However, while remaining members of the group did not go for kotlin. So finally, we went for java.

### **5.2 Three Tier Design**

Deciding how to judiciously divide the project between all team members was another design issue. We finally decided on a 3-tier design, which is an application program organized into three major parts.

The three parts are:

1. The Frontend Part
2. The Backend Part
3. The Database Part

A 3-tier application uses the client/server computing model. With three tiers or parts, each part can be developed concurrently by a different team of programmers. Because the programming for a tier can be changed or relocated without affecting the other tiers, the 3-tier model makes it easier for an enterprise or software packager to continually evolve an application as new needs and opportunities arise. Existing applications or critical parts can be permanently or temporarily retained and encapsulated within the new tier of which it becomes a component.  This design idea was very appealing to our team, especially for portability purposes.

### **5.3 Schedule**

As a team we also chose not to include Machine Learning in our project. It is not easy to implement machine learning to predict homeopathic medicines by reading prescriptions and taking into account the symptoms. It is a very tedious and difficult job to implement specially for beginners like us who started doing their first Software Engineering project. We also recognized that we would not have enough time to properly implement such a feature and decided to exclude it from our plan. We used to work almost every day 1-2 hours daily. And more on the weekends and divided the work among the team members evenly based on their experiences and interests. We plan to bring this into our project in the next version.